Sumário

[Introdução: A Filosofia de Construção do Criatech3D System V-6.0 3](#_Toc211494602)

[Prioridade 1: A Fundação - Gestão de Dependências com Composer 3](#_Toc211494603)

[Prioridade 2: A Eficiência - O Princípio DRY (Don't Repeat Yourself) 3](#_Toc211494604)

[Prioridade 3: A Organização - O Padrão de Roteamento "Front Controller" 4](#_Toc211494605)

[Manual de Criação de Módulos - A Estrutura Padrão (Versão Final) 5](#_Toc211494606)

[O Motivo e a Função de Cada Arquivo 5](#_Toc211494607)

[1. As Rotas: public/index.php 5](#_Toc211494608)

[2. O Controller Principal: controllers 6](#_Toc211494609)

[3. A View da Lista: views/index.php 6](#_Toc211494610)

[4. A View do Formulário: views/form.php 7](#_Toc211494611)

[5. O JavaScript da Lista: public/dist/js/produtos-search.js 7](#_Toc211494612)

[6. O JavaScript do Formulário: public/dist/js/produto-form.js 7](#_Toc211494613)

[7. As APIs dos Modais: public/api/salva-categoria.php 7](#_Toc211494614)

[Manual de Arquitetura por Módulo - Criatech3D System V-6.0 (Versão Final) 8](#_Toc211494615)

[1ª Parte: Módulos Essenciais do Sistema 8](#_Toc211494616)

[Módulo A: Login e Autenticação 8](#_Toc211494617)

[Módulo B: Dashboard 9](#_Toc211494618)

[Módulo C: Componentes Visuais (partials) e o Menu Dinâmico 9](#_Toc211494619)

[2ª Parte: Módulos de Cadastro e Gestão 10](#_Toc211494620)

[Módulo 1: Colaborador (Funcionário) 10](#_Toc211494621)

[Módulo 2: Material 11](#_Toc211494622)

[Módulo 3: Equipes 11](#_Toc211494623)

[3ª Parte: Módulos do Gerenciador (Administração Avançada) 12](#_Toc211494624)

[Módulo 1: Painel de Permissões 13](#_Toc211494625)

[Módulo 2: Controle de Acesso por Horário 13](#_Toc211494626)

[Módulo 3: Liberação de Acesso 14](#_Toc211494627)

[Módulo 4: Perfis Temporários 15](#_Toc211494628)

[Módulo 5: Espelhar Permissões 15](#_Toc211494629)

[4ª Parte: Gestão de Contratos 16](#_Toc211494630)

[Módulo 1: Contrato 16](#_Toc211494631)

[5ª Parte: Módulos de Administração do Sistema 17](#_Toc211494632)

[Módulo 1: Módulos do Sistema 17](#_Toc211494633)

[Módulo App V.1: Arquitetura e Implementação Inicial 18](#_Toc211494634)

[1. Arquitetura de Autenticação Multi-Empresa (Multi-Tenant) 18](#_Toc211494635)

[2. Arquitetura do Aplicativo Flutter: O Padrão de Módulos 18](#_Toc211494636)

[2.1. O Pacote de Arquivos de um Módulo Flutter 19](#_Toc211494637)

# Introdução: A Filosofia de Construção do Criatech3D System V-6.0

Bem-vindo ao manual de manutenção e desenvolvimento do Criatech3D System. O sistema que você tem em mãos é o resultado de um processo deliberado de modernização, projetado não apenas para funcionar hoje, mas para ser uma base sólida, escalável e de fácil manutenção para o futuro.

Se você está a ler isto para criar um novo sistema ou para dar manutenção neste, é crucial que comece por entender os **três pilares arquitetónicos** sobre os quais este projeto foi construído. Estes não são apenas detalhes técnicos; são os princípios que guiarão todo o futuro desenvolvimento.

## Prioridade 1: A Fundação - Gestão de Dependências com Composer

* **O Quê:** O primeiro passo em qualquer projeto PHP moderno é a adoção do **Composer**. Ele foi implementado para gerir o "coração" do sistema: o autoloading de classes.
* **Por Quê:** Antes, era necessário usar require\_once no topo de cada ficheiro para carregar as dependências, uma prática propensa a erros e que polui o código. Com o autoloader do Composer, as classes (FuncionarioController, AuthManager, etc.) são carregadas automaticamente no momento exato em que são necessárias.
* **Diretriz para o Futuro:** Todo novo sistema **deve** começar com o Composer. Ele é a fundação que nos permite construir de forma limpa e, mais importante, nos dá a capacidade de integrar bibliotecas de terceiros (para gerar PDFs, processar pagamentos, etc.) de forma profissional e segura.

## Prioridade 2: A Eficiência - O Princípio DRY (Don't Repeat Yourself)

* **O Quê:** Durante a refatoração, identificámos e eliminámos agressivamente a duplicação de código. Isto foi feito em duas frentes críticas:
  1. **No JavaScript:** A lógica de busca em tempo real, que estava copiada em múltiplos ficheiros (empresas-search.js, contratos-search.js), foi centralizada numa única função reutilizável em public/dist/js/app-utils.js.
  2. **Nas APIs PHP:** A lógica para salvar cadastros simples a partir de janelas modais, que estava espalhada por vários ficheiros salva-\*.php, foi unificada no controllers/ApiController.php.
* **Por Quê:** Código repetido é o inimigo da manutenção. Um bug corrigido num local precisaria de ser corrigido em dez outros. Ao centralizar a lógica, uma única alteração beneficia todo o sistema. Se decidirmos adicionar um ícone de "a carregar" à busca, mudamos num só sítio e todas as listas do sistema são atualizadas.
* **Diretriz para o Futuro:** Antes de copiar e colar um bloco de código, pergunte-se: "Esta lógica pode ser necessária noutro lugar?". Se a resposta for sim, invista o tempo para criar uma função, método ou classe reutilizável. Este é o segredo para construir sistemas que são rápidos de expandir e fáceis de corrigir.

## Prioridade 3: A Organização - O Padrão de Roteamento "Front Controller"

* **O Quê:** Abandonámos a arquitetura onde cada ficheiro .php na pasta /public era uma página. Adotámos o padrão **Front Controller**, onde um único ficheiro, public/index.php, serve como o ponto de entrada para todas as requisições.
* **Por Quê:** Esta é a mudança estrutural mais importante. Um único ponto de entrada, combinado com um Roteador (core/Router.php), oferece benefícios imensos:
  + **Segurança Centralizada:** A verificação de sessão acontece num único local, antes de qualquer outra lógica ser executada.
  + **URLs Amigáveis:** As URLs agora são limpas e lógicas (ex: /projetos/editar/4), em vez de expor a estrutura de ficheiros do sistema.
  + **Organização Clara:** A responsabilidade de cada componente é bem definida. O Roteador direciona, o Controller pensa e a View exibe.
* **Diretriz para o Futuro:** Todo novo sistema deve seguir este padrão. Ele estabelece uma separação clara de responsabilidades que torna o código mais fácil de entender, depurar e escalar.

# Manual de Criação de Módulos - A Estrutura Padrão (Versão Final)

Para criar ou dar manutenção em qualquer módulo no sistema, é preciso entender que ele é composto por um "pacote" de arquivos interligados. Cada arquivo tem uma responsabilidade clara. Usaremos um módulo genérico chamado **"Produto"** como exemplo.

**O Pacote de Arquivos de um Módulo**

Este é o conjunto de arquivos que você precisa para criar um novo módulo chamado "Produto":

| **Componente** | **Arquivo de Exemplo** |
| --- | --- |
| **1. As Rotas (O Ponto de Entrada)** | public/index.php |
| **2. O Controller Principal** | controllers/ProdutoController.php |
| **3. A View da Lista** | views/produto/index.php |
| **4. A View do Formulário** | views/produto/form.php |
| **5. O JavaScript da Lista** | public/dist/js/produtos-search.js |
| **6. O JavaScript do Formulário** | public/dist/js/produto-form.js |
| **7. (Opcional) As APIs dos Modais** | public/api/salva-categoria.php |

## O Motivo e a Função de Cada Arquivo

### 1. As Rotas: public/index.php

* **O Que é?** É o **mapa do sistema**. É o arquivo mais importante, pois atua como o **ponto de entrada único** para todas as requisições.
* **Por Que é Essencial?** Se um link ou formulário não tiver uma rota correspondente definida aqui, o sistema não saberá o que fazer e retornará um erro "404 - Página não encontrada". É a primeira peça que você deve configurar.
* **O Que Ele Faz?** Ele contém a "lista de endereços" que liga uma URL a uma ação específica. Por exemplo:

PHP

// Quando o usuário acede a /produtos, chama o método index() do ProdutoController

**$router->add('GET', '/produtos', [ProdutoController::class, 'index']);**

// Quando o formulário envia dados para /produtos/salvar, chama o método store()

**$router->add('POST', '/produtos/salvar', [ProdutoController::class, 'store']);**

### 2. O Controller Principal: controllers

* **O Que é?** É o **cérebro** do módulo. A rota (index.php) decide qual "cérebro" chamar.
* **O Que Ele Faz?**
  + **index():** Prepara e mostra a página com a lista de todos os produtos.
  + **create():** Prepara e mostra o formulário para adicionar um novo produto.
  + **edit($id):** Busca os dados de um produto específico no banco e mostra o formulário preenchido para edição.
  + **store():** Recebe os dados do formulário quando o usuário clica em "Salvar" e os grava no banco de dados.
  + **delete($id):** Remove um produto do banco de dados.
  + **search():** Atende às chamadas da busca em tempo real, pesquisa no banco e devolve o HTML da lista de resultados.

### 3. A View da Lista: views/index.php

* **O Que é?** É a **vitrine** do módulo. É o arquivo HTML que o Controller manda exibir.
* **O Que Ele Faz?**
  + Contém o HTML do título da página, do botão "Novo Produto" e da estrutura da tabela (o cabeçalho <thead>).
  + O corpo da tabela (<tbody id="produtos-table-body">) é deixado **vazio**. Ele será preenchido dinamicamente pelo JavaScript.
  + Contém o campo de busca (<input id="search">).

### 4. A View do Formulário: views/form.php

* **O Que é?** É a **prancheta de cadastro**. É o arquivo HTML que o Controller usa para as ações de create e edit.
* **O Que Ele Faz?**
  + Contém todos os campos (<input>, <select>, <textarea>) para os dados do produto.
  + Se for um módulo complexo, é aqui que a estrutura de abas é definida.
  + Contém o código HTML das janelas **Modais**.

### 5. O JavaScript da Lista: public/dist/js/produtos-search.js

* **O Que é?** É o **assistente de busca** da página de listagem.
* **O Que Ele Faz?**
  + A sua única responsabilidade é "ligar" a busca em tempo real. Ele diz ao sistema: "Quando alguém digitar no campo id='search', chame a API definida na rota /api/produtos/search e coloque o resultado dentro da tabela com id='produtos-table-body'".

### 6. O JavaScript do Formulário: public/dist/js/produto-form.js

* **O Que é?** É o **cérebro interativo** do formulário.
* **O Que Ele Faz?**
  + Controla toda a interatividade do formulário que não envolve recarregar a página.
  + Faz os botões + abrirem as janelas **Modais**.
  + Contém a lógica para os botões "Salvar" dentro dos Modais, que chamam as APIs de salvamento rápido.

### 7. As APIs dos Modais: public/api/salva-categoria.php

* **O Que é?** É um **serviço de entrega rápida**.
* **O Que Ele Faz?**
  + Recebe os dados de um Modal (ex: o nome da nova categoria).
  + Usa o ApiController para salvar essa informação no banco de dados.
  + Devolve uma resposta em JSON para o produto-form.js, que então atualiza o menu dropdown na tela principal com a nova categoria, sem precisar de recarregar a página.

# Manual de Arquitetura por Módulo - Criatech3D System V-6.0 (Versão Final)

## 1ª Parte: Módulos Essenciais do Sistema

### Módulo A: Login e Autenticação

* **Descrição:** Controla o acesso ao sistema. É o portal de entrada que valida as credenciais do usuário e cria a sua sessão de trabalho.
* **Pacote de Arquivos:**
  + **Rotas (public/index.php):**

PHP

// Exibe o formulário de login

**$router->add('GET', '/login', [AuthController::class, 'showLoginForm']);**

// Processa os dados do formulário de login

**$router->add('POST', '/login', [AuthController::class, 'processLogin']);**

// Encerra a sessão do usuário

**$router->add('GET', '/logout', [AuthController::class, 'logout']);**

* + **Controller Principal:** controllers/AuthController.php
  + **View do Formulário:** views/auth/login.php
  + **Componente de Segurança:** security/AuthManager.php (contém a lógica real de validação)
* **Tabelas do Banco de Dados:**
  + **Principais:** usuarios, sessoes\_ativas
  + **Consultadas:** loginperfil, acesso\_perfil\_horario, acesso\_usuario\_horario

### Módulo B: Dashboard

* **Descrição:** É a página inicial do sistema após o login, servindo como o ponto central de navegação.
* **Pacote de Arquivos:**
  + **Rota (public/index.php):**

PHP

// Exibe a página principal do dashboard

**$router->add('GET', '/dashboard', [DashboardController::class, 'index']);**

* + **Controller Principal:** controllers/DashboardController.php
  + **View Principal:** views/dashboard/index.php
* **Tabelas do Banco de Dados:** Nenhuma diretamente.

### Módulo C: Componentes Visuais (partials) e o Menu Dinâmico

* **Descrição:** Esta não é uma funcionalidade visível, mas sim o "esqueleto" que dá consistência a todas as páginas.
* **Pacote de Arquivos:**
  + views/partials/navbar.php: A barra de navegação superior.
  + views/partials/footer.php: O rodapé.
  + views/partials/sidebar.php: O componente mais importante, o menu lateral.
* **O sidebar.php e o Banco de Dados:** O sidebar.php é o coração da navegação e da segurança visual. Ele **não** tem um menu fixo no código. Em vez disso, ele executa o MenuPermissionManager::getVisibleMenuItems() que faz uma consulta complexa ao banco de dados para construir o menu dinamicamente para o usuário que está logado. Ele lê a tabela modulos para saber o nome, o ícone e a rota de cada item, e cruza essa informação com as tabelas perfil\_modulo\_permissao e usuario\_modulo\_permissao para decidir se o usuário tem ou não permissão para ver aquele link. É por isso que o menu se adapta a cada perfil de usuário.

## 2ª Parte: Módulos de Cadastro e Gestão

### Módulo 1: Colaborador (Funcionário)

* **Descrição:** Módulo complexo para o cadastro completo de colaboradores, organizados em abas.
* **Pacote de Arquivos:**
  + **Rotas (public/index.php):**

PHP

**$router->add('GET', '/funcionario', [FuncionarioController::class, 'index']);**

**$router->add('GET', '/funcionario/novo', [FuncionarioController::class, 'create']);**

**$router->add('GET', '/funcionario/editar/{id}', [FuncionarioController::class, 'edit']);**

**$router->add('POST', '/funcionario/salvar', [FuncionarioController::class, 'store']);**

**$router->add('GET', '/funcionario/excluir/{id}', [FuncionarioController::class, 'delete']);**

**$router->add('GET', '/api/funcionarios/search', [FuncionarioController::class, 'search']);**

* + **Controller Principal:** controllers/FuncionarioController.php
  + **View da Lista:** views/funcionario/index.php
  + **View do Formulário:** views/funcionario/form.php
  + **JavaScript:** dist/js/funcionarios-search.js, dist/js/funcionario-form.js
  + **APIs dos Modais:** api/salva-perfil.php, api/salva-cidade.php, api/get-cidades.php
* **Tabelas do Banco de Dados:**
  + **Principais:** funcionario, usuarios, enderecos\_pessoas, loginperfil
  + **Consultadas:** perfil, estados, cidades

### Módulo 2: Material

* **Descrição:** Módulo de cadastro simples, servindo como template para entidades que não possuem relacionamentos complexos nem interface com abas.
* **Pacote de Arquivos:**
  + **Rotas (public/index.php):**

PHP

**$router->add('GET', '/material', [MaterialController::class, 'index']);**

**$router->add('GET', '/material/novo', [MaterialController::class, 'create']);**

**$router->add('GET', '/material/editar/{id}', [MaterialController::class, 'edit']);**

**$router->add('POST', '/material/salvar', [MaterialController::class, 'store']);**

**$router->add('GET', '/material/excluir/{id}', [MaterialController::class, 'delete']);**

**$router->add('GET', '/api/materiais/search', [MaterialController::class, 'search']);**

* + **Controller Principal:** controllers/MaterialController.php
  + **View da Lista:** views/material/index.php
  + **View do Formulário:** views/material/form.php
  + **JavaScript da Lista:** public/dist/js/materiais-search.js
* **Tabelas do Banco de Dados:**
  + **Principal:** p\_material
  + **Consultada:** medidas

### Módulo 3: Equipes

* **Descrição:** Módulo essencial para a organização dos recursos humanos de campo. Permite a criação de grupos de trabalho (equipas), a designação de um líder e a atribuição de múltiplos colaboradores (usuários) a cada equipa. Este módulo serve de base para a atribuição de projetos.
* **Pacote de Arquivos:**
  + **Rotas (public/index.php):**

PHP

**$router->add('GET', '/equipes', [EquipeController::class, 'index']);**

**$router->add('GET', '/equipes/novo', [EquipeController::class, 'create']);**

**$router->add('GET', '/equipes/editar/{id}', [EquipeController::class, 'edit']);**

**$router->add('POST', '/equipes/salvar', [EquipeController::class, 'store']);**

**$router->add('GET', '/equipes/excluir/{id}', [EquipeController::class, 'delete']);**

**$router->add('GET', '/api/equipes/search', [EquipeController::class, 'search']);**

* + **Controller Principal:** controllers/EquipeController.php
  + **View da Lista:** views/equipe/index.php
  + **View do Formulário:** views/equipe/form.php
  + **JavaScript:** public/dist/js/equipes-search.js, public/dist/js/equipe-form.js
* **Tabelas do Banco de Dados:**
  + **Principais:** equipes, equipe\_membros
  + **Consultadas:** usuarios, perfil
* **Relações e Fluxo de Negócio:**
  + **Dependências:** Para criar uma equipa, é necessário que existam Colaboradores (usuários) previamente cadastrados. A seleção de líderes é filtrada para excluir perfis de "equipe".
  + **Regras de Negócio Chave:** O formulário impede que um mesmo colaborador seja atribuído a mais de uma equipa, destacando visualmente os membros que já estão alocados. Permite a filtragem de membros por perfil para facilitar a montagem da equipa.

## 3ª Parte: Módulos do Gerenciador (Administração Avançada)

O menu "Gerenciador" agrupa várias funcionalidades de administração de alto nível, todas controladas principalmente pelo GerenteController.php.

### Módulo 1: Painel de Permissões

* **Descrição:** O centro de controlo para as permissões do sistema. Permite definir a visibilidade e as ações para perfis e usuários específicos através de uma interface de abas.
* **Pacote de Arquivos:**
  + **Rotas (public/index.php):**

PHP

**$router->add('GET', '/gerente', [GerenteController::class, 'index']);**

**$router->add('POST', '/gerente/salvar-visibilidade-perfil', [GerenteController::class, 'storeVisibilidadePerfil']);**

**$router->add('POST', '/gerente/salvar-acoes-perfil', [GerenteController::class, 'storeAcoesPerfil']);**

**$router->add('POST', '/gerente/salvar-visibilidade-usuario', [GerenteController::class, 'storeVisibilidadeUsuario']);**

**$router->add('POST', '/gerente/salvar-acoes-usuario', [GerenteController::class, 'storeAcoesUsuario']);**

* + **Controller Principal:** controllers/GerenteController.php
  + **View Principal (Container):** views/gerente/index.php
  + **Views das Abas:** views/gerente/aba\_visualizacao.php, aba\_acoes.php, aba\_visualizacao\_usuario.php, aba\_acoes\_usuario.php
  + **JavaScript:** public/dist/js/gerente-permissoes.js
* **Tabelas do Banco de Dados:**
  + **Leitura e Escrita:** perfil\_modulo\_permissao, perfil\_acao\_permissao, usuario\_modulo\_permissao, usuario\_acao\_permissao
  + **Consultadas:** usuarios, perfil, modulos

### Módulo 2: Controle de Acesso por Horário

* **Descrição:** Permite definir os dias da semana e horários em que cada perfil ou usuário pode aceder ao sistema.
* **Pacote de Arquivos:**
  + **Rotas (public/index.php):**

PHP

**$router->add('GET', '/controle-acesso', [ControleAcessoController::class, 'index']);**

**$router->add('POST', '/controle-acesso/salvar-regras-perfil', [ControleAcessoController::class, 'storeRegrasPerfil']);**

**$router->add('POST', '/controle-acesso/salvar-regras-usuario', [ControleAcessoController::class, 'storeRegrasUsuario']);**

* + **Controller Principal:** controllers/ControleAcessoController.php
  + **View Principal:** views/controle\_acesso/index.php
  + **Componente Reutilizável:** views/partials/tabela\_horarios.php
* **Tabelas do Banco de Dados:**
  + **Leitura e Escrita:** acesso\_perfil\_horario, acesso\_usuario\_horario

### Módulo 3: Liberação de Acesso

* **Descrição:** Mostra uma lista de todos os usuários logados no sistema em tempo real e permite que um administrador encerre a sessão de um usuário à força.
* **Pacote de Arquivos:**
  + **Rotas (public/index.php):**

PHP

**$router->add('GET', '/liberacao\_acesso', [LiberacaoAcessoController::class, 'index']);**

**$router->add('GET', '/liberacao-acesso/liberar/{id}', [LiberacaoAcessoController::class, 'release']);**

* + **Controller Principal:** controllers/LiberacaoAcessoController.php
  + **View Principal:** views/liberacao\_acesso/index.php
  + **JavaScript:** public/dist/js/liberacao-acesso-search.js
* **Tabelas do Banco de Dados:**
  + **Leitura e Escrita:** sessoes\_ativas

### Módulo 4: Perfis Temporários

* **Descrição:** Permite conceder um perfil a um usuário por um período determinado (com data de início e fim).
* **Pacote de Arquivos:**
  + **Rotas (public/index.php):**

PHP

**$router->add('GET', '/perfis\_temporarios', [PerfilTemporarioController::class, 'index']);**

**$router->add('POST', '/perfis\_temporarios/salvar', [PerfilTemporarioController::class, 'store']);**

**$router->add('GET', '/perfis\_temporarios/excluir/{id}', [PerfilTemporarioController::class, 'delete']);**

* + **Controller Principal:** controllers/PerfilTemporarioController.php
  + **View Principal:** views/perfis\_temporarios/index.php
* **Tabelas do Banco de Dados:**
  + **Principal:** usuario\_perfil\_temporario
  + **Consultadas:** usuarios, perfil

### Módulo 5: Espelhar Permissões

* **Descrição:** Ferramenta para copiar todas as permissões personalizadas de um usuário de origem para um usuário de destino.
* **Pacote de Arquivos:**
  + **Rotas (public/index.php):**

PHP

**$router->add('GET', '/espelhar\_permissoes', [GerenteController::class, 'showEspelharPermissoesForm']);**

**$router->add('POST', '/espelhar\_permissoes/salvar', [GerenteController::class, 'storeEspelhamento']);**

* + **Controller Principal:** controllers/GerenteController.php
  + **View Principal:** views/gerente/espelhar\_permissoes.php
* **Tabelas do Banco de Dados:**
  + **Leitura e Escrita:** usuario\_modulo\_permissao, usuario\_acao\_permissao

## 4ª Parte: Gestão de Contratos

### Módulo 1: Contrato

**Descrição:** Módulo para formalizar e gerir os contratos com as empresas. Um contrato vincula uma Empresa a um Serviço específico e detalha os prazos. A sua interface em abas permite a associação de múltiplas equipes de trabalho ao contrato.

**Pacote de Arquivos:**

* **Rotas (public/index.php):**

PHP

**$router->add('GET', '/contratos', [ContratoController::class, 'index']);**

**$router->add('GET', '/contratos/novo', [ContratoController::class, 'create']);**

**$router->add('GET', '/contratos/editar/{id}', [ContratoController::class, 'edit']);**

**$router->add('POST', '/contratos/salvar', [ContratoController::class, 'store']);**

**$router->add('GET', '/contratos/excluir/{id}', [ContratoController::class, 'delete']);**

**$router->add('GET', '/api/contratos/search', [ContratoController::class, 'search']);**

* **Controller Principal:** controllers/ContratoController.php
* **View da Lista:** views/contrato/index.php
* **View do Formulário:** views/contrato/form.php
* **JavaScript:** dist/js/contrato-search.js, dist/js/contrato-form.js
* **APIs dos Modais:** api/salva-servico.php, api/salva-orgao.php (setor), api/salva-tipo-equipe.php
* **Tabelas do Banco de Dados:**
  + **Principais:** contrato, contrato\_equipes
  + **Consultadas:** empresas, servicos, setor, tipo\_equipes

## 5ª Parte: Módulos de Administração do Sistema

Esta seção cobre os módulos "meta", ou seja, aqueles que servem para administrar o próprio funcionamento e estrutura do sistema.

### Módulo 1: Módulos do Sistema

**Descrição:** Ferramenta de administração para gerir os próprios módulos da aplicação. Permite criar, editar e definir a hierarquia (menus e submenus), rotas, ícones e, crucialmente, as permissões de acesso base para cada perfil de usuário. É o painel de controle que define a estrutura de navegação do sistema.

**Pacote de Arquivos:**

* **Rotas (public/index.php):**

PHP

**$router->add('GET', '/modulos', [ModuloController::class, 'index']);**

**$router->add('GET', '/modulos/novo', [ModuloController::class, 'create']);**

**$router->add('GET', '/modulos/editar/{id}', [ModuloController::class, 'edit']);**

**$router->add('POST', '/modulos/salvar', [ModuloController::class, 'store']);**

**$router->add('GET', '/modulos/excluir/{id}', [ModuloController::class, 'delete']);**

**$router->add('GET', '/api/modulos/search', [ModuloController::class, 'search']);**

* **Controller Principal:** controllers/ModuloController.php
* **View da Lista:** views/modulos/index.php
* **View do Formulário:** views/modulos/form.php
* **JavaScript:** public/dist/js/modulos-search.js
* **Tabelas do Banco de Dados:**
  + **Principais:** modulos, perfil\_modulo\_permissao
  + **Consultadas:** perfil

# Módulo App V.1: Arquitetura e Implementação Inicial

**Introdução**

Este documento detalha a arquitetura e os passos de implementação da terceira fase do Criatech3D System: o aplicativo móvel. Ele serve como um mapa de desenvolvimento, conectando as decisões de negócio descritas nos documentos de requisitos (Projeto APP.docx e App.docx) com a implementação técnica.

O objetivo principal do aplicativo é estender a funcionalidade do sistema web para as equipas de campo, permitindo a gestão de projetos, validação de tarefas e execução de Ordens de Serviço (OS) de forma remota, com capacidade de funcionamento offline.

## 1. Arquitetura de Autenticação Multi-Empresa (Multi-Tenant)

## 2. Arquitetura do Aplicativo Flutter: O Padrão de Módulos

Para garantir a escalabilidade e a organização do aplicativo, espelhando a filosofia do backend, adotámos uma arquitetura modular. Em vez de agrupar os arquivos por tipo (ex: todas as telas numa única pasta), nós agrupamo-los por funcionalidade. Cada módulo é uma pasta autocontida com tudo o que é necessário para a sua função: modelos de dados, telas, lógica e componentes.

Esta abordagem segue diretamente os princípios de Organização (Prioridade 3) e Eficiência (DRY - Prioridade 2) definidos para o sistema web.

* Estrutura de Pastas Principal (lib/):
* lib/
* ├── api/
* │ └── api\_service.dart # Camada de acesso à rede (HTTP)
* ├── modules/ # Pasta raiz para todos os módulos funcionais
* │ ├── auth/ # Módulo de Autenticação
* │ └── projetos/ # Módulo de Projetos
* └── main.dart # Ponto de entrada da aplicação

### 2.1. O Pacote de Arquivos de um Módulo Flutter

Cada módulo dentro da pasta modules/ terá uma estrutura interna consistente, similar ao "pacote de arquivos" dos módulos PHP.

| Componente | Pasta/Arquivo de Exemplo (Módulo Projetos) | Responsabilidade | Equivalente no PHP |
| --- | --- | --- | --- |
| Modelos (Models) | projetos/models/projeto\_model.dart | Define a estrutura dos dados (o "molde") para os objetos do módulo. | Arrays de dados do Controller |
| Telas (Screens) | projetos/screens/projetos\_list\_screen.dart | Constrói a interface visual principal do módulo. | View (ex: views/produto/index.php) |
| Widgets | projetos/widgets/projeto\_list\_item.dart | Pedaços reutilizáveis da interface (ex: um card). | Partials (ex: views/partials/) |
| Lógica (Providers) | projetos/providers/projetos\_provider.dart | O "cérebro" do módulo. Busca dados, gere o estado e notifica as Telas. | Controller (ex: ProdutoController.php) |